COL780 Computer Vision
Assignment-3

Somanshu Singla 2018EE10314
Lakshya Kumar Tangri 2018EE10222

November 10, 2021

Pedestrian Detection

Contents

1 Overview of Models Used

We have implemented these two algorithms using OpenCV and Pytorch for our tasks:

1. *HOG based SVM detector*[1]:
   - Sample P positive samples from your training data of the object(s) you want to detect and extract HOG descriptors from these samples
   - Sample N negative samples from a negative training set that does not contain any of the objects you want to detect and extract HOG
   - Train a Linear/RBF Support Vector Machine on your positive and negative samples.
2. *Faster RCNN*[2]: Faster R-CNN is a deep convolutional network used for object detection, that appears to the user as a single, end-to-end, unified network. The network can accurately and quickly predict the locations of different objects.

   A Faster R-CNN object detection network is composed of a feature extraction network which is typically a pretrained CNN, similar to what we had used for its predecessor. This is then followed by two subnetworks which are trainable. The first is a Region Proposal Network (RPN), which is, as its name suggests, used to generate object proposals and the second is used to predict the actual class of the object.

2 Overview of Pre Processing Techniques Used

We have used the below mentioned techniques in our model:

1. *Filtering*: We have used Bilateral Filtering to reduce the noise in input images.
2. *Image Pyramid*: We have used the concept of Image Pyramid so as to detect objects at multiple scales [3]

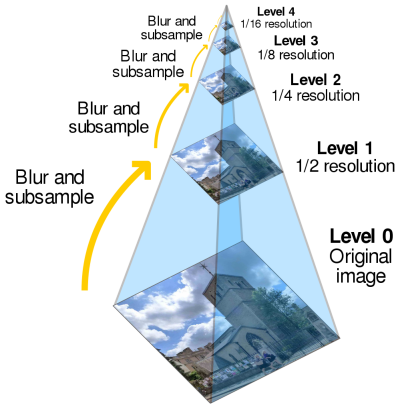


Figure 1: Image Pyramid [4]

3 HOG based SVM

3.1 Pretrained Model

We have implemented a HoG person detector using *cv2.HOGDescriptor()* and *setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())* given in OpenCV. This detector is based on [1].

3.1.1 Results

- Average Precision: 0.0546
- Average Recall @1: 0.0528
- Average Recall @10: 0.125

3.2 Custom Model Implementation Details

In the HOG feature descriptor, the distribution (histograms) of directions of gradients (oriented gradients) are used as features. Gradients (x and y derivatives) of an image are useful because the magnitude of gradients is large around edges and corners (regions of abrupt intensity changes) and we know that edges and corners pack in a lot more information about object shape than flat regions.

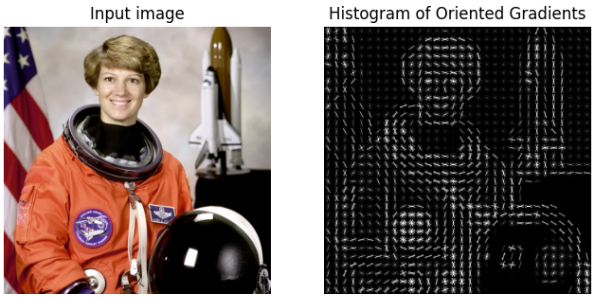


Figure 2: HOG of a person (Source: link)

The Complete process which we follow in this part is:

1. We train the model as per the steps mentioned in overview section.
2. During prediction for each image, we take it's pyramidal representation. Then for each level in the image pyramid we use sliding window approach.
3. For each sliding window, the trained classifier makes it's prediction.
4. For all windows which are predicted as pedestrian, appropriate changes to height and width are made according to the scale of image.
5. Non-Maximal Suppression[5] is used to remove some of the predictions

3.2.1 Results

The trained SVM model can be found: here

- Average Precision: 0.0427
- Average Recall @1: 0.0686
- Average Recall @10: 0.124

4 Faster RCNN

4.1 Implementation Details

We have implemented a Faster RCNN person detector using a pretrained model on COCO dataset from *torchvision* library. We implemented the complete inference pipeline using dataloaders, NMS[5]. This detector is based on [2].

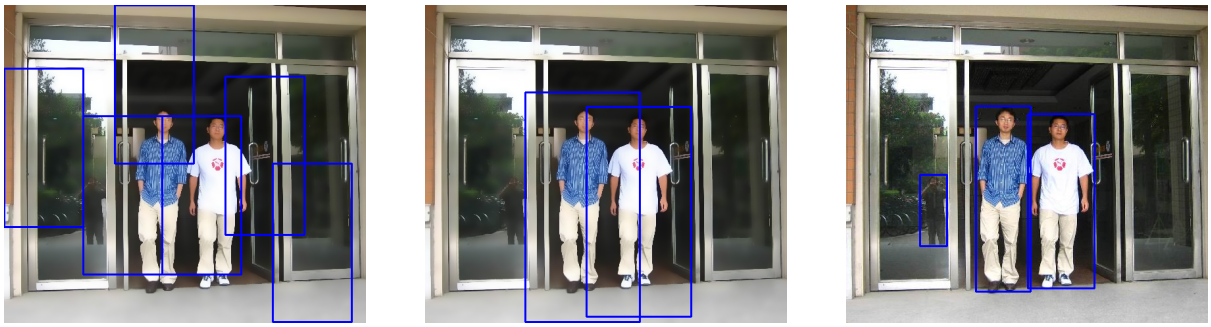
4.2 Results

The weights file of pretrained Faster RCNN model can be found: here

- Average Precision: 0.750
- Average Recall @1: 0.303
- Average Recall @10: 0.803

5 Difference in Performance

In terms of the precision and recall metrics the clear order of performance is: Faster RCNN > Pretrained HoG based SVM > Custom HoG based SVM.
Some examples[1] where the predictions of these models differ:

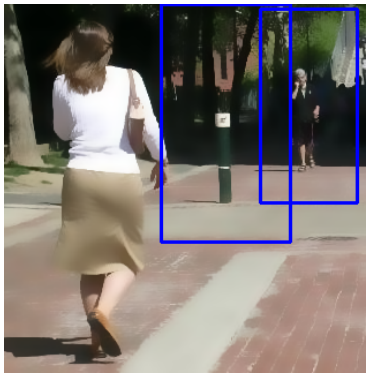


(a) Custom HoG (b) Pretrained HoG (c) Faster RCNN

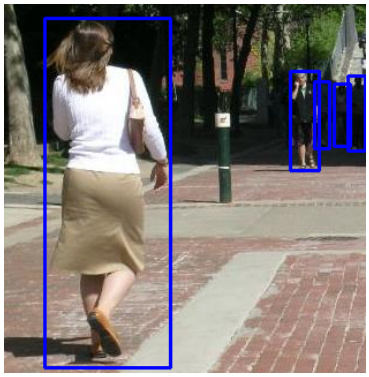
Figure 3: Example-1

[1] In all the examples blue bounding boxe show the prediction of the model

(a) Custom HoG

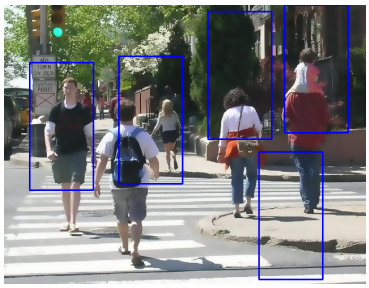


(b) Pretrained HoG

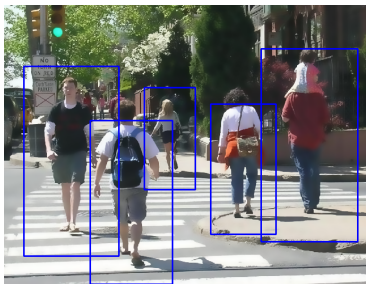


(c) Faster RCNN

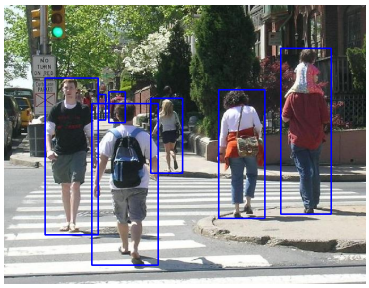
Figure 4: Example-2



(a) Custom HoG

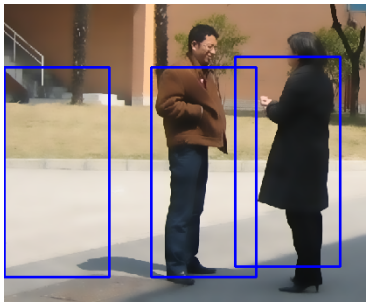


(b) Pretrained HoG

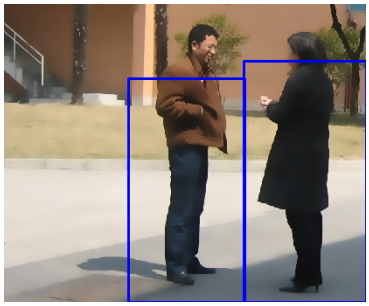


(c) Faster RCNN

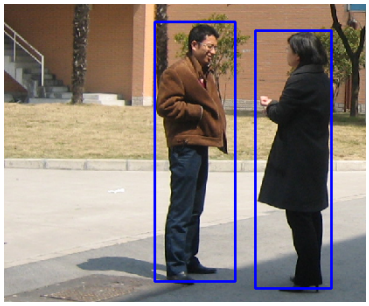
Figure 5: Example-3



(a) Custom HoG



(b) Pretrained HoG



(c) Faster RCNN

Figure 6: Example-4

(a) Custom HoG (b) Pretrained HoG (c) Faster RCNN

Figure 7: Example-5

As we saw in the metrics that the difference between Custom HoG and Pretrained HoG is small the same can be observed by 6 and 4 in one case the Custom HoG makes a bad prediction and in one case the Pretrained HoG makes a bad prediction. In all the examples the Faster-RCNN makes the best predictions,it predicts the highest number of pedestrians and makes minimum errors.

Some reasons behind this might be:

- As Pretrained and Custom HoG models are based on the same idea and algorithms their similar performance is what we would have expected
- Faster RCNN is known to be a much better model, which is also visible in the results. Faster RCNN model uses Neural Networks which have been shown capable of learning otherwise difficult problems.

References

[1] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, pp. 886–893 vol. 1, 2005.

[2] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems* (C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, eds.), vol. 28, Curran Associates, Inc., 2015.

[3] G. F. Geeks, "Image pyramid using opencv — python." Available at https://www.geeksforgeeks.org/image-pyramid-using-opencv-python/.

[4] Wikipedia, "Pyramid (image processing)." Available at https://en.wikipedia.org/wiki/Pyramid_(image_processing).

[5] PyImageSearch, "Nms." Available at https://github.com/PyImageSearch/imutils.